LOCATION INTELLIGENCE SERVICES FOR MOBILES USING RUBY ON RAILS AND JQUERYMOBILE

Alfio Costanzo, Alberto Faro and Concetto Spampinato
Department of Electrical, Electronics and Computer Engineering, University of Catania, Catania, Italy

Keywords: Location Intelligence, Fuzzy Systems, Semantic Web, Location based Services, Mobile Computing.

Abstract: The current applications for mobiles provide information about Location Based Services (LBS) of user interest that don't take into account the current status of the services neither they take into account the real time conditions of the traffic network or of the weather. Aim of the paper is to show how location intelligence services may be implemented by an open source solution that is able to help the activities and to support the decisions of the mobile users taking advantage from all the information collected by the sensors located on the field and from the business data stored on the disparate data stores that may be of interest of the urban/metropolitan mobility, security and logistics. Fuzzy logic and semantic web technologies are taken into account to improve the current LBS applications. A case study developed using Ruby on Rails and JQueryMobile illustrates how such a web service may work in practice.

1 INTRODUCTION

The current applications for mobiles provide information about Location Based Services (LBS) of user interest that don't take into account the current status of the services neither they take into account the current conditions of the traffic network or of the weather. Overcoming such limits is not easy since this needs the availability of information outside the control of the service providers, such as the travel time to reach the destination depending on the current traffic flows, the current availability of parking vacancies and so on.

Also, the mobile applications don't support m-commerce, i.e., commercial transactions carried out on-line by mobile users. Implementing real time and m-commerce services is only the first step to activate advanced LBSs. In fact, it is also important to provide the users with the information that best fits their needs. For example, a cheaper park may be suggested with lower priority if there is another park that is more close to the destination in case of raining. Another step is the one of integrating the administrative Data Bases (DBs) resident on separate computers relevant for LBS so that the user may be informed on all the services potentially available at the urban/metropolitan scale.

For this reason, the future mobile application should be powered more and more by artificial intelligence techniques, thus giving rise to location intelligence services of practical user interest (Faro, 2011a). Although many intelligent mobile services may be theoretically offered to the users, as the ones proposed in the field of the intelligent transportation systems (ITS) (McCubbin, 2003), they are rarely offered in practice to the users. Often, the services offered are not conceived to support the user decisions or to optimize the user activities but to provide the users with average information that give a moderate help to their activities.

Moreover, ITS services are mainly distributed by proprietary solutions that don't take advantage from the information resident on the disparate DBs of potential user interest. For example, sudden traffic congestions or dangerous situations are not reported to the interested users, service reservation cannot be carried out on-line, neither the users are advised when products of their interest are available on some store.

Aim of the paper is to show how location intelligence services may be implemented by an open source solution that is able to help decisions and activities of the mobile users taking advantage from all the information collected in the data stores that may be of interest of the urban/metropolitan mobility.

In the paper we assume that the area is provided with sensors that monitor in real time the traffic

Costanzo A., Faro A. and Spampinato C..
LOCATION INTELLIGENCE SERVICES FOR MOBILES USING RUBY ON RAILS AND JQUERYMOBILE.
DOI: 10.5220/0003940707630771
In *Proceedings of the 8th International Conference on Web Information Systems and Technologies* (WEBIST-2012), pages 763-771
ISBN: 978-989-8565-08-2

conditions and the weather. Also, we assume that business information about availability, vacancies and cost of urban/metropolitan services are stored in some server connected to internet.

Thus, the paper does not deal with the basic monitoring technologies widely studied in the literature, e.g., (Faro, 2008, 2011b)], but it is specifically devoted to illustrate how to integrate all the relevant information available on the net to support the main use cases of the mobile users depending on their current position. Fuzzy logic rules and metadata technologies are used to help the user mobility and her/his mobile business activities.

The Ruby on Rails (RoR) (Hartl, 2011) framework is adopted to develop the web service since it allows the designer to organize the application as a collection of use cases that can be reused for similar tasks (Faro, 1998, 2033a). Moreover, RoR is provided with a powerful language, i.e., Ruby, that facilitates the implementation of: a) the fuzzy rules that address user mobility and aid their decisions, and b) the procedures to access the metadata layer that integrate the disparate DBs. Other two languages may be also used in RoR to facilitate the implementation of LBS applications: a) Java scripts to exchange with mobiles information geo-referenced on Google Maps, and b) JQueryMobile (Bai, 2011) to convey such information in a user friendly format that may be visualized, without any modification, on PCs, tablets and mobiles.

Section 2 shows how simple fuzzy logic and Horn Logic rules may be implemented by Javascript and JQueryMobile to improve the location based services. Section 3 discusses how the use of JQueryMobile and Javascript allow us to use the metadata technology to favour the integration of the proprietary DBs of interest of mobile users. The advantages of implementing the Location Intelligence services by using RoR and JQuery Mobile will be discussed in section 4 by a small case study dealing with a prototypical web application, called WiCity, currently under test at our University, that illustrates how an user can connect her/his mobile to an RoR server to be informed by a suitable interface, developed by JQueryMobile, on some basic LBSs concerning mobility (parks, gas stations and traffic congestions), health services (pharmacies and first aid services), events/places of tourist interest, and on the routes to reach by car the chosen destination from the current user position by taking into account the current traffic flows and weather conditions. When discussing this prototype we will outline how the methodological issues pointed out in the previous sections may be used to implement effective location intelligence services for citizens and tourists.

2 IMPROVING LBS BY FUZZY AND HORN CLAUSES

Location based services may be considered as a sort of generalization of the Intelligent Transportation Systems. The latter are mainly dedicated to improve mobility and logistics activities of the mobile users, the former aim at supporting such activities taking into account also security, commerce and business requisites. Thus, in LBS environment it is important to reach the destination in the minimum time, but also to avoid accidents or congested areas. Analogously, the user may decide to follow some non minimal path to reach the destination if this is done in more safe conditions depending on the weather conditions. As well as, the user may be interested in paying the parks depending on the real parking time rather than paying in advance basing on some forecast of this time.

Although the rules that support such LBS requirements are very simple, the current LBS applications are mainly conceived as information systems that provide the users with general information while they are walking or driving. Thus, in the following we will show how the LBS effectiveness may be improved by using suitable fuzzy logic rules (Wang, 2001) and Horn clauses expressed in Prolog (Wielemaker, 2009).

2.1 Nearest Services and Safe Walking

To help the users to choose the most suitable nearest services we should return to them a Google Map on her/his mobile that shows the current user position and two circles, one, let say C_w, with a radius of few hundreds of meters and the second, let say C_c, with a radius of about one kilometre, containing the markers of the services located in such areas. The former circle should point out the services that are reachable by walking, whereas the second should point out the ones reachable by car.

In principle, one can define the circle radius following a very simple rule, i.e., the services located in the smaller circle are recommended to the walking people if the distance *dist* from the current user position is such that *dist* < 300 meters. This constraint would become *dist* < 1000 meters if the user is driving a car.

However, if there are no services available within such circles, the user might accept a moderately greater distance to find services of interest. Of course, an arbitrary modification of such distance is not acceptable, whereas it is fair to inform the user on how much a certain increase of the circle radius may causes a corresponding decrease of her/his satisfaction of the solution provided.

The computation of the radius corresponding to the user expectations is straightforward in the fuzzy logic framework in case we have to consider only one condition that may influence the notion of *nearest*. Indeed, for example, assuming that the rule is "*if the user is not young, then the user would like to have the required service very close*", and that the user is 28 years old, from the fuzzy sets in fig.1c we have that the evidence that she/he is not young is given by the membership *not* [μ_{young}(28 years old)], i.e., 0.8, and consequently the most suitable radius is the x-coordinate of the barycentre of the area M1 in fig.1a, i.e., 175 meters.

The computation of the best radius *d* is a few more complicated if we wish to take into account more conditions together, plus the current user position. In fact, the RoR application should be provided with the fuzzy set of each condition and with an algorithm to combine the radii derived from the various conditions. To show how this can be done, let us assume that our rule is: "*if the user is not young* ***and*** *it is cloudy, then the user would like to have the required service very close*". In this case, if the people is 28 years and the sky is partially cloudy (e.g., cloudiness degree = 0.4), we may derive the maximum distance as the x-coordinate of the barycentre of the two masses M1 and M2 in fig.1a. The former represents the distance to be suggested considering only the age, the latter refers only to the cloudiness. Approximately it is 190 mt. Of course, if the *and* contained in the rule is substituted by *or*, the radius decreases since it is the x-coordinate of the barycentre of the mass M1 or M2 that has the greater membership, i.e., about 175 mt.

Thus, to solve the above problem, the RoR application should know the current time and weekday, as well as the user age and health status, and the traffic and weather conditions measured by dedicated sensing infrastructures.

How to measure the travel times for each street has been widely analyzed by the authors in other papers, whereas, in the case study, we will show how the current weather conditions may be obtained by connecting the RoR web application to Yahoo.

For what concerns the walking time to destination we may assume that it depends on the distance between the current user position and the destination as suggested by Google Maps. But, in case there are dangerous areas to avoid, the RoR web application should inform the users about alternative paths to reach either the destination or safer locations. However, indicating alternative paths using Google Maps is not a trivial job since the routes suggested by Google Maps are based on average conditions that cannot be modified easily by the programmer. A solution of this problem is the one of displaying the alternative pedestrian routes like the minimum time driving routes using Google Maps but in the more elaborated way discussed in the next section.

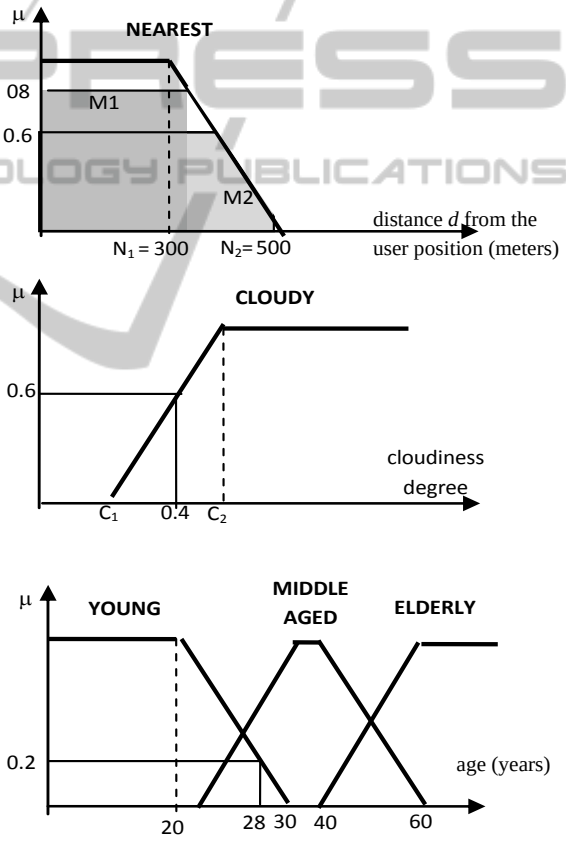

Figure 1: Fuzzy sets associated with the words: nearest, young, middle aged, elderly, and cloudiness, where μ [0,1] is the membership degree to such words of the values of the definition domain, e.g., μ_{young}(28 years old) = 0.2.

2.2 Safe and Fast Driving in the Traffic

The computation of the minimal route connecting two intersections is certainly instrumental to find the best route between two addresses. It may be

obtained by any program that is able to find the best path connecting two points of a graph (Kumari, 2010). In (Faro, 2011a) we have suggested a Mobility Based Prolog (MBP) Program, since it may be used with little modifications to find also other routes of interest of the user as the ones to find the service that are nearest to the destination.

Also, this MBP program shows very satisfactory time performance and allows us to update the travel time of a street between two adjacent intersections x, y from to t_1 to t_2 by simply retracting the *fact* travel_time (x, y, t_1) and asserting the new *fact* travel_time(x, y, t_2) in the Prolog knowledge base. In this way, any accident or congestion or work in progress involving a street may be immediately taken into account by the program.

Another advantage of using Prolog is that a simple generalization of the MBP program allows us to solve the most relevant logistics problem, i.e., the one to compute the minimal Hamiltonian cycle of the graph associated to the traffic net, i.e. the cycle that starting from an address will come back to the initial address in a minimum time by traversing a set of prefixed intermediate points.

Moreover, the previous fuzzy rules may be easily written in Prolog, thus making possible that the MBP program uses the membership values to express if the routes or the mentioned Hamiltonian cycles found are very (enough/few) fast (fluent/slow). As a consequence, if the user is interested only to fast routes to destination, the program may suggest the route only if $\mu_{fast}(T_{route}) > 0.8$, otherwise the user is invited to postpone her/his travel.

2.3 Displaying the Best Routes

A suitable graphical interface that is compatible with the one used frequently by the user is very important for the usability of the web service (Giordano, 2002). For this reason we aim at informing the user on the best path to destination or on the best cycle to distribute or collect goods, by representing the traffic network as a graph superimposed to the Google Maps in such a way that its arcs coincide with the streets and the nodes with the street intersections.

Thus, after having identified the triples travel_time (x, y, t) for every adjacent intersection pair and having found the best route by using he mentioned MBP program we should draw on Google Maps the best path from a source intersection S to the destination intersection D by the subsequent drawing of linear traits connecting the adjacent nodes traversed during the path. However, the lines of such drawing don't correspond necessarily to those of Google Maps since the links between nodes are not always linear segments, neither we have a database containing the adjacent nodes with their geographical coordinates. Thus, we have two problems: a) to find the geo-coordinates of all the intersections, and b) to draw the links between adjacent nodes like the ones of Goole Maps.

To solve the first problem it would be enough to use the function "x AT y" that gives the coordinates of the intersection between the road x and y. But, this function is available for US, and not for all the countries. Thus, our first problem is to find an alternative way to compute all the intersection geo-coordinates in the urban/metropolitan area. Fortunately, this can be accomplished as follows: a) pass to the API *Directions* of Google Maps the names of the pair of streets (x, y) that have an intersection to find the *route* that allows us to reach by *walking* the initial address of x from the last address of y (or in some case the initial address of x to the initial address of y), and b) extract the geographical coordinates of the first marker that contains in its info window the name of the route y, thus finding the geo-coordinates of the intersection.

However, as pointed out above, the knowledge of the intersections together with their geo/coordinates is not enough to find the optimal route from any source address ad_s to any destination address ad_d. Indeed, to solve the problem we have to execute the following further tasks:

- to find the adjacent intersections. This can be obtained by using the API *Directions* by verifying for each pair of intersections if they are connected by one step link, and
- to compute for any address ad_r the set $\boldsymbol{AD}_{\mathbf{out}}$ (ad_s) of intersections that can be reached in one step from ad_s and the set $\boldsymbol{AD}_{\mathbf{in}}$ **(**ad_d**)** of intersections that allow us to reach in one step ad_d. These sets can be computed by using *Directions* to find the intersections around ad_s that can be reached by driving in one step from ad_s, and the intersections around ad_d that can be reached by walking in one step from ad_d. Of course, we have to exclude in the latter case the routes not allowed to drivers.

Finally, we have to compute, e.g., by using the mentioned MBP program, all the routes connecting any intersection in $\boldsymbol{AD}_{\mathbf{out}}$ to any intersection in $\boldsymbol{AD}_{\mathbf{in}}$. The best route is the one that is obtained by minimizing, for any intersection belonging to $\boldsymbol{AD}_{\mathbf{out}}$ (ad_s) and to $\boldsymbol{AD}_{\mathbf{in}}$ (ad_d), the travel time T consisting of the following three terms:

$$T = t(ad_s, \boldsymbol{AD}_{\mathbf{out}}(ad_s)) + t(\boldsymbol{AD}_{\mathbf{out}}(ad_s), \boldsymbol{AD}_{\mathbf{in}}(ad_d)) + t(\boldsymbol{AD}_{\mathbf{in}}(ad_d), ad_d) \quad (1)$$

To draw the same intersection links that will be drawn by Google Maps, we use again *Directions* to draw the connections between the adjacent intersections of the best route by simply requiring that such route is obtained by using repeatedly *Directions* to draw the best route between any pair of adjacent nodes belonging to the best route.

3 DATA INTEGRATION USING OWL-LIKE METADATA

The use of proprietary DBs is still convenient today to manage the data warehouse of any organization. But, simple commercial transactions and pure information tasks push more and more for the use of standard formats, such as RDF based DBs (Powers, 2003), especially in the LBS framework. Indeed, the data stores based on OWL, that is an extension of RDF, or even on XML favour the integration of data belonging to different organizations. For example, the 'public' part of the data of an organization could be mapped in OWL and sent to a central server where such data will be available for all the users through a standard interface.

Alternatively, the XML/OWL data could remain on the servers of the organizations if the central server is able to carry out distributed queries to collect the data useful for the mobile user. This will favour the updating of the data and the system reliability.

The data, in standard format, could be also stored on the mobiles, even if this solution is suitable only for data that are few dependent on time, otherwise their frequent updating may interfere with the normal operations of the mobile. The technologies available on the market allows us to implement all the above solutions not only to support the centralized or the distributed access to the DBs, but also to facilitate the mapping of the relational DBs to triple stores, or the production of novel OWL DBs from scratch, e.g., (Allemang, 2011), (Bonomi, 2007), (Faro, 2003b) and (Zhai, 2008). For example, software environments such as Protègè (Knubblauch, 2004) are suitable to design novel RDF stores, whereas servers such as Sesame (Broekstra, 2002) may be used to implement centralized RDF stores available to web users. The use of JQueryMobile facilitates the access to the DBs from mobiles either directly or with the help of the central information server (David, 2011).

Developing the information server as an RoR application allows us to design the web service as a collection of use cases. This will improve the verification, the test and the maintenance of the software especially when the work flow of the application is complicated for the presence of several cooperating actors.

4 CASE STUDY

This case study illustrates how we connect a mobile to an RoR web application, called WiCity, that includes all the technical issues discussed in the previous sections. The interested reader may download the software from code.google.com/p/query-mobile-unict/source/browse/.

Fig.2 shows the WiCity architecture, where the user mobiles are connected to a central RoR server. Currently, all the needed information is stored in the MySQL tables of the RoR server. We are also developing a distributed version of WiCity where the RoR server will use both local databases and the remote XML/OWL data stores resident on the proper remote servers (i.e., Sesame server). This architecture will favour data integration, data privacy and data updating according to the methods outlined in sect.3.

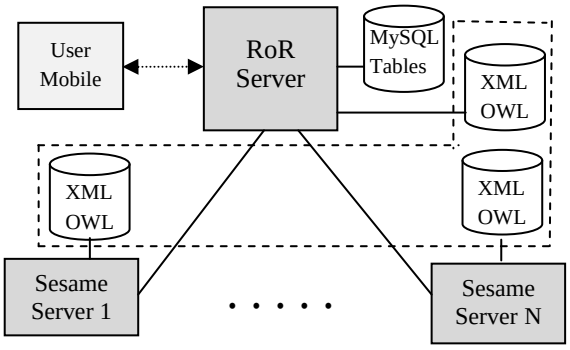

Figure 2: Current centralized WiCity architecture. It is conceived to support future integrations of XML/OWL data stored on different remote servers.

Fig.2 points out that, currently, the user mobiles may use only data stored on the RoR server, whereas, in the future remote data may be used too. In particular, data integration will be obtained by connecting local and remote XML/OWL data stores through messages exchanges between the web RoR application stored on the RoR server and the Sesame servers at distant service points.

The mobile interface, developed by JQueryMobile, currently consists of some icons

concerning information on mobility and health services, and on the best routes to reach the destination from the current user position (fig.3.left). Also, events/places of general interest are provided to tourists and citizens (fig.3.right).

Figure 3: The interface icons of WiCity are chosen from a predefined list: main interface (on the left) and events of general interest (on the right).

The *events of general interest* consist of icons chosen by the user from the list of available LBSs. In particular, the one on the left_top in fig.3.rigth is connected to an RoR process implemented on the server that is able to access the weather metadata of Yahoo. After having processed the JSON file received from Yahoo such process extracts the weather conditions of the area in which the mobile is located, i.e., cloudy and 17 °C in fig.4.left.

Such data are not only useful to inform the user on the current weather conditions but also to modify the fuzzy sets associated to the concept *nearest service*. The basic mobility and heath services (e.g., parks, gas stations, pharmacies and first aid points) may be accessed either in alphabetical order to get relevant information such as address, location on the map, opening hours and so on (see fig.4 right), or by the mentioned fuzzy facilities.

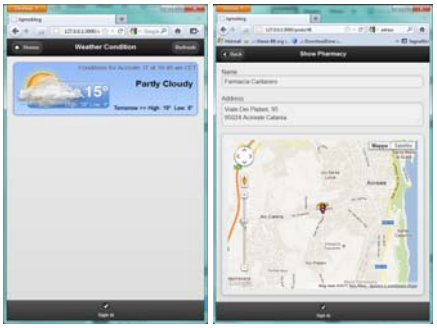

Figure 4: Weather info and localization of a pharmacy of user interest chosen from a predefined list.

Currently, only fuzzy facility that take into account the current traffic and weather conditions are available; they are denoted as *services nearest to my current position*. For example, fig.5 shows how WiCity points out the pharmacies that may be reached by walking or by car displaying the corresponding markers within a circle of a suitable radius obtained by using the mentioned fuzzy rules.

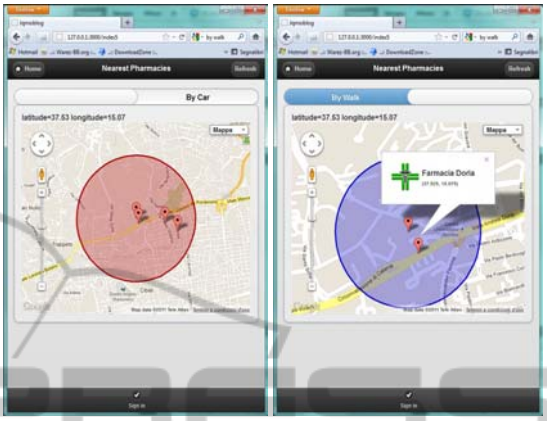

Figure 5: Pharmacies that may be reached by car (on the left) and by walking (on the right) taking into account the weather info and the user position.

To use the facilities of WiCity, any user should be registered. In the information related to her/his profile, WiCity includes automatically the items inserted by the user so that they are available for the other users of her/his community as shown in fig.6. In this ways we should obtain two benefits:

- to encourage the users to insert information useful for their community, and
- to avoid that they insert deliberately wrong information.

The use of the registered information to provide the users with e-government and e-commerce services (e.g., certificates, event tickets, etc.) is for future works.

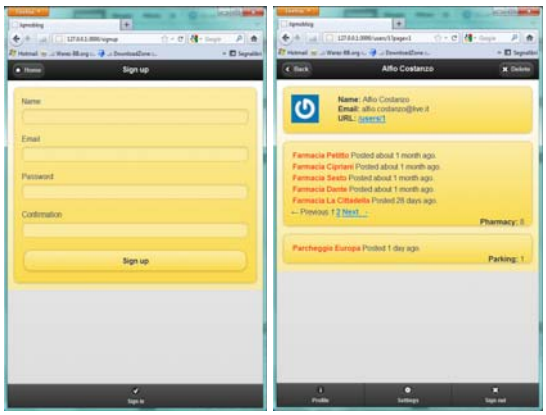

Figure 6: Registration form (on the left) and list of the items inserted by the user (on the rigth).

In fig.7 we show the WiCity mobile interface obtained by using Flash Builder by Adobe (Gassner, 2010). The effort for developing this version of WiCity is certainly lower than the one needed to develop an RoR application powered by JQueryMobile, but the organization of the data is not so effective as the one supported by RoR that is based on the well known paradigm Models-Views-Controllers (Hartl, 20011).

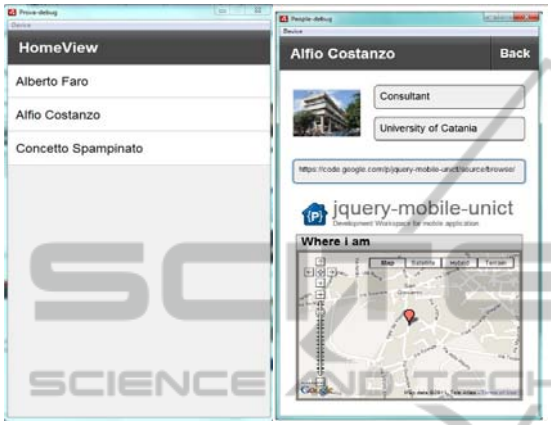

Figure 7: Some WiCity snapshots in Flash Builder.

One positive feature of the Flash Builder applications is that it may use XML-like files, as the ones outlined in fig.8, stored locally or on a remote server.

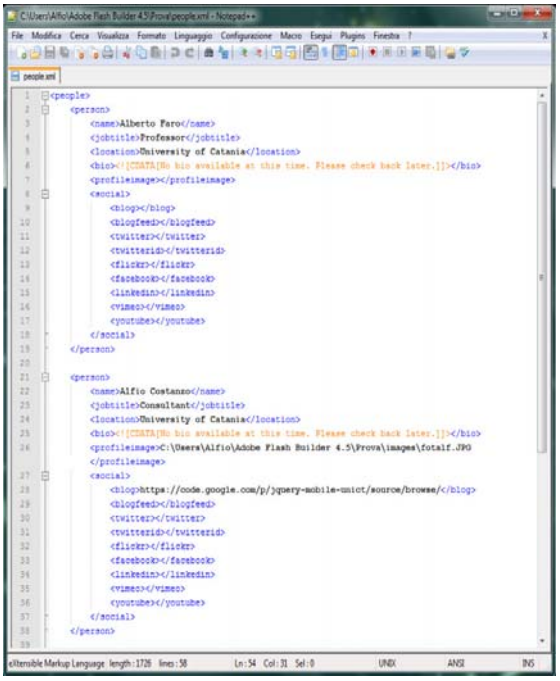

Figure 8: The XML file used by the Flash Builder application installed on the user mobile.

A more powerful facility of using metadata may be obtained by using the RoR framework. As an example, WiCity makes possible the geo-localization of the services by geo-markers whose info windows are filled with the data extracted from XML/OWL metadata as shown in fig.9 and fig.10.

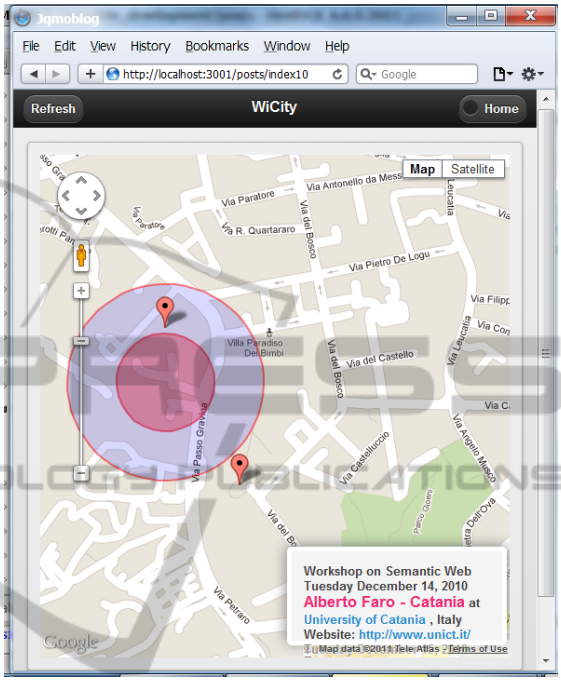

Figure 9: Geo-localization of points that are reachable by walking and by car. These points are related to the event described in the window on the bottom. The data of the event are taken from the metadata.

Finally, let us clarify in fig.11 how the *Directions* API of Google Maps is useful to find the intersections between two routes x and y in order to obtain the function (x AT y) currently not available for many countries.

Indeed, in fig.11.left the intersection between the road named 'costarelli' and the one named 'del toscano' is not pointed out by using *Directions* to connect 'costarelli' and 'del toscano' with the option *by driving*, whereas it is pointed out when it is executed with the option *by walking*. Indeed, often there is no intersection between two routes by using *Directions* with the option *by driving*, since as shown in fig.11.left, *Directions* takes into account the one way streets. On the contrary, the information contained in the info window of the function *Directions* with the option *by walking* (e.g., fig.11right) allows us to discover the marker and related geo-coordinates associated to the intersection between the roads.

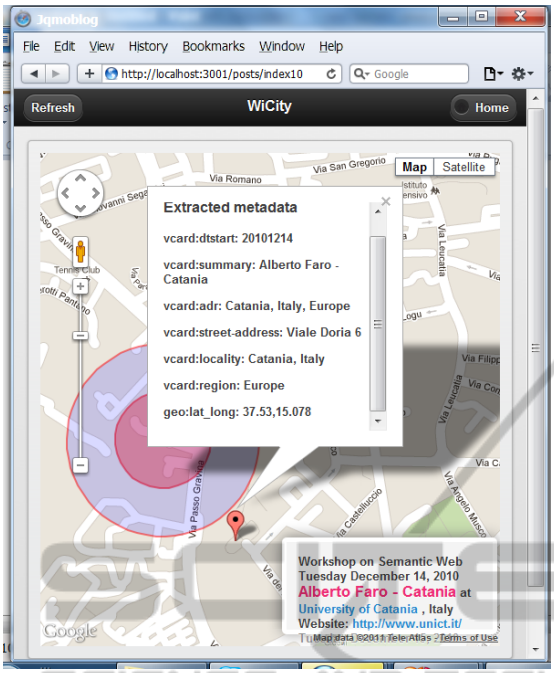

Figure 10: The info window associated to the points of interest is filled with the data extracted from the metadata.

How to obtain all the intersections and to derive the matrix that gives the adjacent intersections of each traffic network intersection is outside the scope of the paper and will be discussed in detail in future works. Once such matrix is obtained, the methodology proposed in sect.2.3 will allow us to display on the user mobiles the current best route to destination.

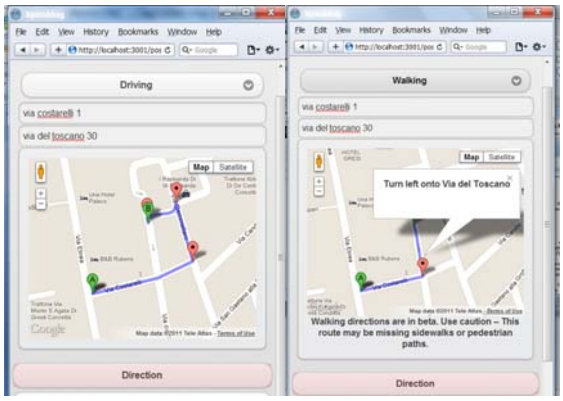

Figure 11: Route connecting the initial and final addresses of two streets in search of the marker associated to their intersection by driving (on the left) and by walking (on the right).

5 CONCLUSIONS

The paper has widely discussed how the current LBS applications may be improved by using fuzzy rules and semantic technologies. The solutions proposed would have the expected high impact since the information will be based on real time updated data stores.

Another expected positive feature of the proposed web services is that the chosen implementation approach favours the integration of the disparate database available at metropolitan scale thus opening concrete opportunities to develop mobile-commerce and mobile-business applications of wider utility.

Further studies are planned to evaluate *how much* location intelligence should be embedded in the LBS applications to really improve the location services offered to the users.

Issues like decision support systems based on fast data mining techniques (Faro, 2011c) and fast image pre-processing, e.g., (Cannavò, 2006) (Crisafi, 2008), for supporting people recognition and people flow control in case of emergency should take advantage from the possibilities of having JQueryMobile based PDAs that are able to suggest timely convenient alternatives in both security and logistics fields to mobile users.

The applications presented in the paper are currently under development within a project called K-Metropolis supported by our Region to favour the transition towards the knowledge society with the aim of improving the level of competitiveness of the local economic system.

Further applications of the proposed technologies are also planned at our University to control physical processes that may influence the people security and the environmental quality such as control systems that alert the drivers on the overflowing of a river by indicating suitable escape routes, or emergency systems that inform timely the policeman in case a high pollution is affecting a certain area of the sea (Spampinato, 2010).

REFERENCES

Allemang D., Hendler, J., 2011. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*, Elsevier Ltd, Oxford, 2011

Bai G., 2011. JQueryMobile. Packt Publishing, 2011

Bonomi A., Rondelli A., Vizzari G., Stride S.., 2007. An Ontology Driven Web Site and its Application in the Archaeological Context, *2nd International Workshop*

on Ontology, Conceptualization and Epistemology for Software and System Engineering, 2007

Broekstra J. et al., 2002. Sesame: a generic architecture for storing and querying RDF and RDF Schema. *LNCS* Vol.2342, 2002.

David M., 2011. Developing Websites with JQueryMobile. Focal Press, 2011

Cannavò F., Nunnari G., Giordano D., Spampinato C., 2006. Variational Method for Image Denoising by Distributed Genetic Algorithms on GRID Environment. Proc. Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE '06. IEEE, 2006

Crisafi A., Giordano D., Spampinato C., 2008. GRIPLAB 1.0: Grid Image Processing Laboratory for Distributed Machine Vision Applications. *Proc. Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE '08*, IEEE, 2008

Faro, A., Giordano, D., 1998. StoryNet: an Evolving Network of Cases to Learn Information Systems Design. *IEE Proceedings SOFTWARE*, 145(4), 119-127, 1998

Faro, A., Giordano, D., 2003. Design memories as evolutionary systems: socio-technical architecture and genetics, Proc. *IEEE International Conference on Systems Man and Cybernetics*, Vol.5, 4334-4339, IEEE, 2003

Faro, A, Giordano, D. and Musarra, A., 2003. Ontology based intelligent mobility systems. Proc. *IEEE Conference on Systems, Man and Cybernetics*, Washington, Vol.5, 4288-4293, IEEE , 2003.

Faro, A., Giordano, D., Spampinato, C., 2008. Evaluation of the traffic parameters in a metropolitan area by fusing visual perceptions and CNN processing of webcam images. *IEEE Transactions on Neural Networks*, vol.19(6), 1108-1129, IEEE, 2008

Faro, A., Giordano, D., Spampinato, C., 2011. Integrating location tracking, traffic monitoring and semantics in a layered ITS architecture, *Intelligent Transport Systems, IET*, Vol.5(3), 197-206, IET, 2011

Faro, A., Giordano, D. and Spampinato, C., 2011. Adaptive background modeling integrated with luminosity sensors and occlusion processing for reliable vehicle detection. *IEEE Transactions on Intelligent Transportation Systems*. Vol. 12(4), 1398-1412, IEEE, 2011.

Faro, A., Giordano, D., Maiorana, F., 2011. Mining massive datasets by an unsupervised parallel clustering on a grid: Novel algorithms and case study. *Future Generation Computer Systems*, Vol. 27(6), 711–724, 2011.

Giordano, D., 2002. Evolution of interactive graphical representations into a design language: a distributed cognition account, *International Journal of Human-Computer Studies*, Vol. 57(4), 317-345, 2002

McCubbin, R. P. Staples, B. L. and Mercer, M. R. 2003. Intelligent Transportation Systems Benefits and Costs: 2003 Update, *Mitretek Systems, Inc.*, Washington.

Gassner D., 2010. Flash Builder 4. Wiley, 2010

Hartl M., 2011. Ruby on Rails 3, Addison Wesley, 2011.

Knubblauch H. et al., 2004. Protege OWL Plugin (an open development environment for semantic web applications. *Proc. ISWC 2004*, *LNCS* Vol.3298, 229-243, 2004

Kumari, S. M. and Geethanjali, N., 2010. A Survey on Shortest Path Routing Algorithms for Public Transport Travel, *Global Journal of Computer Science and Technology*, Vol. 9(5), 2010.

Powers,S. 2003 Practical RDF, O'Reilly Media

Spampinato, C., Giordano, D., Di Salvo, R., Chen-Burger, Y. H. J., Fisher, R. B., Nadarajan, G., 2010. Automatic fish classification for underwater species behavior understanding, in Proceedings of the *first ACM international workshop on Analysis and retrieval of tracked events and motion in imagery streams, ARTEMIS '10*, ACM, 45-50, 2010.

Wang P. P., 2001. Computing with words. Wiley Inderscience, 2001

Wielemaker, J. Hildebrand, M. and Ossenbruggen, J. 2009. Using Prolog as the fundament for applications on the semantic web, *hcs.science.uva.nl/projects/SWI-Prolog/ articles/mn9c.pdf*, 2009.

Zhai, J., Jiang, J., Yu, Y, and Li, J., 2008. Ontology-based Integrated Information Platform for Digital City, *IEEE Proc. Of Wireless Communications, Networking and Mobile Computing*, WiCOM '08, IEEE, 2008.